1489

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 17/22, 17/27	A1	(11) International Publication Number: WO 96/17310 (43) International Publication Date: 6 June 1996 (06.06.96)

(21) International Application Number: PCT/US95/15266

(22) International Filing Date: 29 November 1995 (29.11.95)

(30) Priority Data:
08/346,476 29 November 1994 (29.11.94) US

(71) Applicant: AVALANCHE DEVELOPMENT COMPANY [US/US]; Suite 100, 4999 Pearl East Circle, Boulder, CO 80301 (US).

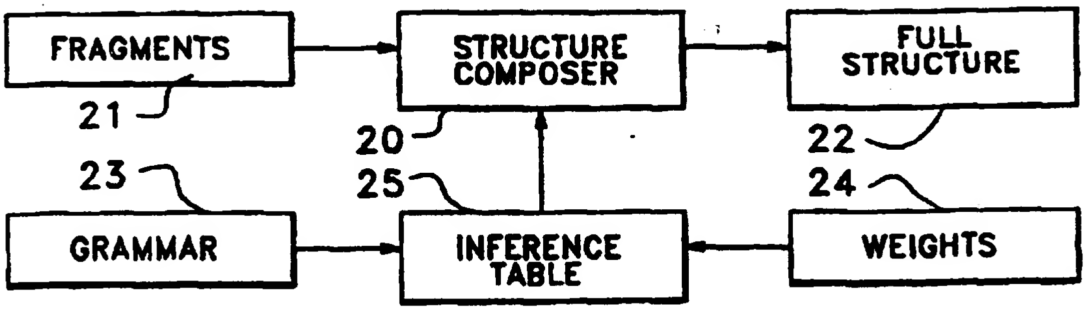
(72) Inventors: OLANDER, Daryl, Brian; 4106 Spy Glass Lane, Niwot, CO 80503 (US). FIFE, Lee, Douglas; 7733 Hygiene Road, Hygiene, CO 80533 (US).

(74) Agents: HANCOCK, Earl, C. et al.; Holland & Hart, Suite 2900, 555 17th Street, Boulder, CO 80201 (US).

(81) Designated States: European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.

(54) Title: SYSTEM AND PROCESS FOR CREATING STRUCTURED DOCUMENTS



(57) Abstract

Many applications require that documents conform to rigorously defined structures. Users define these structures by providing a set of rules, or a grammar (23), to which the structure of documents must conform in order to be acceptable. These rules are entered into the system. The user then enters fragments (21) of the document together with indicators of the intended structural role for the fragments into the system. The document fragments (21) and indicators are inspected and compared against the original grammar (23) to determine what structural modifications and extensions (20) must be made to fit the document fragments together into a single document conforming to the specified grammar. A conforming document is constructed thereby fragment by fragment.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria
AU	Australia
BB	Barbados
BE	Belgium
BF	Burkina Faso
BG	Bulgaria
BJ	Benin
BR	Brazil
BY	Belarus
CA	Canada
CF	Central African Republic
CG	Congo
CH	Switzerland
CI	Côte d'Ivoire
CM	Cameroon
CN	China
CS	Czechoslovakia
CZ	Czech Republic
DE	Germany
DK	Denmark
ES	Spain
FI	Finland
FR	France
GA	Gabon
GB	United Kingdom
GE	Georgia
GN	Guinea
GR	Greece
HU	Hungary
IE	Ireland
IT	Italy
JP	Japan
KE	Kenya
KG	Kyrgystan
KP	Democratic People's Republic of Korea
KR	Republic of Korea
KZ	Kazakhstan
LI	Liechtenstein
LK	Sri Lanka
LU	Luxembourg
LV	Latvia
MC	Monaco
MD	Republic of Moldova
MG	Madagascar
ML	Mali
MN	Mongolia
MR	Mauritania
MW	Malawi
NE	Niger
NL	Netherlands
NO	Norway
NZ	New Zealand
PL	Poland
PT	Portugal
RO	Romania
RU	Russian Federation
SD	Sudan
SE	Sweden
SI	Slovenia
SK	Slovakia
SN	Senegal
TD	Chad
TG	Togo
TJ	Tajikistan
TT	Trinidad and Tobago
UA	Ukraine
US	United States of America
UZ	Uzbekistan
VN	Viet Nam

SYSTEM AND PROCESS FOR CREATING STRUCTURED DOCUMENTS

BACKGROUND OF THE INVENTION

Field of the Invention:

The present invention relates to apparatus and methods for organizing structured information in response to incomplete or fragmented input data. More particularly, the present invention relates to apparatus and processes especially useful in the production of organized text in response to input information provided by a user.

Description of the Prior Art:

Communication effectively contains two types of information. The primary level is the information content itself, whereas the secondary level is information concerning that content. For example, in spoken conversation, the actual words are the first level content while the manner of speaking, such as whispering, shouting, normal inflection, etc., is the second level providing additional information about the meaning of the spoken words. In typical printed matter, the first level again consists of the words making up the printed matter, while the second level of information specifies how the words look: what fonts are used, if certain words are printed in boldface or italic, etc.

Historically, printed documents have used certain typographic conventions to represent the logical structure of the document. For example, certain words may be displayed using a bold font to indicate that those words convey particularly important information. Similarly, certain parts of documents may be displayed differently to indicate that they have a different role in the structure of the document. For example, one might render the title of this section "Description of the Prior Art" using a different font than the body of the section. This would indicate two things: first that the text "Description of the Prior Art" is the title of the section, and more subtly that a new section begins with the title and continues through the body.

This example illustrates how logical structure is made up both of the actual objects found in a document (the title and the body paragraphs), as well as container objects which are only implicitly present in the document (the section implied by the presence of a section title).

A variety of different approaches have been used to indicate both the typographic effects intended when a document is printed or otherwise displayed and the logical structure intended by such a document. Historically, when a document was to be printed, a typesetter was provided with a combination of text for printing and notations called markup which were indicative of special instructions to the typesetter. These markup notations told the typesetter exactly how to print the text: e.g. what type size to use, what kind of type face to use (e.g. bold, italic, etc.), what indentations to use, etc.

However, there are two problems with this historical approach. First, there was a great variety of markup schemes used as well as of typesetting equipment used. This made it difficult to exchange documents between various environments while preserving the intended typographic effects. Second, the logical structure of such a document using formatting-based markup is only implied by the markup. In our example above, there is no explicit marking that a section is present in the document. Instead, a human reader has to infer the presence of the section by using the typographic clues.

To address these problems, a number of different approaches to markup have been used. These approaches are all characterized by using "logical" markup: in this case, a document consists again of the actual content of the document together with markup instructions. With logical markup however, the markup instructions indicate the structure of the document and the structural role of the parts of the document, rather than the typographic effects to be applied to the parts of the document. Continuing with our example, there might be a markup instruction immediately before the title "Description of the Prior Art" indicating that a new section was beginning (such instructions are called "start tags" because they tag the start of a new portion of the document). Following the section start tag, might be another start tag indicating that a title begins at that point. The text of the title would then follow. After the title, another markup instruction would follow indicating that the title had ended (called an end tag). After the title end tag, all the paragraphs of the section body follow. Before each paragraph would be a paragraph start tag and following each paragraph would be a paragraph end tag. Finally, following all the paragraphs in the section would be a section end tag explicitly marking that the section had ended. The following illustrates markup for such a logical structure.

```
<Start of Section>
     <Start of Title>
          Description of the Prior Art
     <End of Title>
     <Start of Para>
          Text of the first paragraph in the
```

```
section
     <End of Para>
     <Start of Para>
          Text of the second paragraph
     <End of Para>
               ...
  <End of Section>
```

Notable among the various approaches relating to logical markup is the Standard Generalized Markup Language (SGML) which is a standard published by the International Organization for Standardization (ISO). SGML is defined by the standard known as ISO 8879:1986. Other approaches to logical markup include the GML language from IBM, and the markup language used by FrameBuilder from Frame Corp. These markup schemes are increasingly used for describing printed documents and are becoming the norm for describing documents and other information intended to be distributed electronically.

Another common feature of logical markup systems is the use of a set of rules, or grammar, to describe the allowed logical structures for a class of documents. By analogy to a grammar for a language which constrains the set of allowed sentences, a grammar for a document constrains the set of allowed logical structures. For example, a particular grammar might require that all documents in the class it describes must begin with a title followed by some number of chapters. The grammar might further require that chapters begin with a chapter title followed by some number of sections. Similarly, sections must begin with a section title followed by some number of paragraphs.

Clearly, different classes of documents have different rules for their logical structures. For example, the logical structure of a legal brief is quite different from that of a novel or that of the documentation for a software product. When the rules for these different classes of documents are collected together, each class will be described by a different grammar.

As the use of logical markup increases, the issue of how such documents are created becomes more important. Typically, the documents are created in one of several ways. They may be created by a human manually inserting the logical markup instructions directly using a word processor. Or they may be inserted by a program which follows certain conventions allowing it to replace formatting-based markup instructions with logical markup instructions. Finally, a human may use a special kind of word processor known as a "structure editor" which allows the human to directly manipulate the structure of the document, by performing actions such as creating a new section and then creating paragraphs within that section. The structure editor then stores the document in a file with the appropriate logical markup.

Each of these techniques has shortcomings. Requiring a human to manually insert all of the logical markup places a burden on the human user: not only does the user need to insert all the logical markup, but the user must have an intimate and complete understanding of the details of the logical structure being created as well as an explicit understanding of the structure they are creating. Using a program to replace formatting-based markup with logical markup is difficult since many of the logical constructs (such as the section in our example above) are only implied by the formatting-based markup and must be inferred. Using a structure editor typically requires the user to create the structure of the document first before entering the content. For example, the user must create a section object and then a paragraph object inside the section before the user can type any text into the paragraph. This does not match the way people work and again places an additional burden on the user.

To address these shortcomings, a technique must be developed which allows the logical structure of a document to be inferred and made explicit via logical markup based on the structure of smaller pieces, or fragments, of the document. For example, if a rule in the grammar for our documents requires that sections consist of an initial title followed by some number of paragraphs, when we see a new title, we can infer that a section begins before that title and insert the start tag for the section. Further, if another section had already begun before this title, we know that other section must finish before the new section can begin and so we can insert an end tag for the first section.

To date, there have been no successful solutions to this problem that are able to cover the entire range of grammars describing classes of allowed document logical structures. Most systems have put the burden on the human user in one of the ways described above.

One notable attempt to solve this problem was previously made by Avalanche Development Company and released in an IBM product known as Text Tagger. Text

Tagger contained a piece of software known as the Document Constructor. The Document Constructor was able to take a specification for a document grammar in a language known as DocSpec and a series of fragments of a document and in some cases construct a legal logical structure which could contain those fragments. The Document Constructor had some serious limitations, however.

First, the input to the Document Constructor was always created manually by a human who understood the intended grammar for the documents and then created a DocSpec description of this grammar. While doing this, the human needed to transform the grammar as required by the limitations of the DocSpec language, in particular to remove any cycles (defined below) from the grammar

Next, the DocSpec language could not describe cyclic grammars. A cyclic grammar is a grammar which describes a logical structure where an element in the structure can either directly or indirectly contain another element of the same type. For example, the rules describing nested lists can either be cyclic or not. A non-cyclic set of rules for nested lists can say that a 1st level list consists of some number of 1st level list items. 1st level list items can be described as consisting of paragraphs together with 2nd las many different types of list as will ever appear nested in a document.

This same rule can be more simply described using a cyclic grammar. In this case, a list can be described as consisting of list items, no matter the level of nesting at which it occurs. List items can be described as consisting of paragraphs together with lists. This set of rules is cyclic since a list can indirectly occur inside another list (by occurring inside a list item). See Figures 4 and 5. Figure 4 shows a cyclic set of rules for lists and Figure 5 displays these rules as a graph where the cycles are easily seen. DocSpec and the Document Constructor could not process grammars containing such cyclic rules. In typical documents, such rules are very common.

Third, the Document Constructor had no way to understand the context of an object, or to qualify the object. These two terms, context and qualification refer to the ability to indicate for example not only that a particular object is a title, but that it is a title in a section (the section provides a context for the title). Thus, in order for a user of Text Tagger to process a grammar where a particular element such as a title could occur in multiple contexts (e.g. titles in sections as well as titles in chapters), the user must rewrite the grammar such that the same element does not occur in multiple contexts (e.g. by creating a new element sect-title to represent section titles and a new element chap-title to represent chapter titles).

Finally, the solutions inferred by the Document Constructor were not necessarily optimal. In this sense, an optimal solution is one that matches a human's intuitive understanding of the structure implicit in a set of fragments and is characterized by a minimal set of structural changes around the fragments, that is by using the smallest possible amount of extra structure around the fragments. The

Document Constructor's inference processing was constructed from a collection of poorly understood and inaccurate heuristics and was not grounded in a rigorous analysis of the problem. This caused its solutions to often be unoptimal and sometimes quite surprising. The error recovery was hand written. It actually occurred outside the Document Constructor itself and consisted of an array of C code.

SUMMARY OF THE INVENTION

Briefly, the present invention is a computer implemented process and system for organizing and composing data, typically document-type data, into a collection that conforms to a set of rules or grammar describing a class of allowed logical structures for such collections. The grammar for the class can either be selected from prestored data or entered by the user using a specification language for such grammars. The process uses the rules for the grammar to construct a set of tables and relationships between the tables used in processing the document. The content of the document is then presented to the process which analyzes the content using the constructed tables to determine an allowed logical structure that can accommodate the content. The content can be a mixture of actual data together with partial and incomplete logical markup. The process completes, reorganizes and assembles this data and partial logical markup to create a document with fully specified structure.

Importantly, the process can apply to any kind of data, whether textual, graphical, or of some other kind, whose logical structure can be described by particular type of grammar known as a context-free grammar amenable to LL(k) parsing together with a set of context-dependent exceptions. Further, the data can be provided to the process in many ways, including as part of a batch process where the entire content is available before the process begins and as part of an interactive process where a human user inputs and alters the content as the process runs. Finally, the completion of the document is minimal in the sense that it makes the smallest number of changes to the fragments and creates the smallest amount of structure around the fragments in order to fit the fragments into a structure allowed by the specified grammar.

The invention is implemented as a process running on a digital computer. Typically, it is embedded in another process. The containing process communicates with the user to obtain the specification of the grammar for the data and to collect the document containing the partial and incomplete markup. The containing process then provides the grammar to the table building portion of the process implementing the invention. When the tables are constructed, the containing process feeds the incomplete document to the process implementing the invention and the invention completes and corrects the logical structure of the document. As each portion of the structure is completed, the process implementing the invention returns the newly completed portion of the document to the containing process.

The containing process may have many forms. It may be a word processor which allows users to interactively enter documents and have their structure completed as part of the editing process. It may also be a batch process which accepts a complete document (although the logical markup of the document is partial) and returns another complete document with the logical structure completed.

The method of this invention is suitable for implementation in a computer environment for composing fragments of input data having partial markup and qualification information associated therewith into a structured organization. Grammar rules defining the outline for the structured organization are selected and stored as tables and indices useful during structure completion from those selected grammar rules. Then each input data fragment is inspected for the minimal modifications of structure within and around the received fragment as is necessary for that fragment to match with the stored grammar rules. The logical structure is then completed within and around that fragment in accordance with the results of the inspecting step. Thus a complete logical structure that is valid in accordance with the selected grammar rules is produced.

The aforementioned inspection can include the step of discarding input data fragments having markup and qualification information associated therewith which are not compatible with the selected grammar rules. The aforementioned completing step can further include the step of producing a complete document incorporating all received fragments which validly conform to the selected grammar rules. The inspection step can still further include the step of inspecting each received input data fragment, and determining the minimal set of structural modifications within and around that fragment to the extent necessary for it to match the stored grammar rules.

Preferably, tables useful in conjunction with the inspection step are formulated with those tables based upon selection of a context free grammar along with context dependent exceptions, and establishment of a set of correction weights. The table building includes the construction of tables and indices including a first segment containing all possible derivation relations between symbols in the selected grammar, a second segment identifying minimum cost symbols that satisfy any given production, and a third portion specifying the minimum cost production sequence that allows derivation of a symbol from another symbol.

The present invention is suitable for implementation in a computer environment for handling fragments of input data having partial markup and qualification information associated therewith and intended for composition into a structured organization determined by selected grammar rules which are stored as tables and indices useful during structure completion based upon those selected grammar rules. The input data fragments are received with their associated markup and qualification information and inspected to determine the minimal set of structural modifications within and around that fragment to the extent necessary for it to match the stored grammar rules. This determination is obtainable by conducting a breadth-first, cost-driven search from the symbols in the fragment to current document structures defined by the selected grammar rules through the space of all possible structures.

Those having normal skill in the art will recognize the foregoing and other objects, features, advantages and applications of the present invention from the following more detailed description of the preferred embodiments as illustrated in the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIGURE 1 is a general block diagram of a computer system environment suitable for implementing the present invention.

FIGURE 2 is a broad illustration of the interrelationships of the elements and process steps in accordance with the present invention.

FIGURE 3 is a more detailed diagram of the Fig. 2 elements and processes.

FIGURE 4 is a typical cyclic set of rules for lists.

FIGURE 5 is a graphical representation of the cyclic rules of Fig. 4.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

A typical computer system environment for implementing the present invention is presented in Fig.

1. Computer 10 employs a conventional central processing unit 12 which functions with memory and/or data storage unit 14 containing a block 15 designated for operation in conjunction with software implementing the present invention. This memory block 15 contains segments dedicated to receiving various content segments 15A through 15N as is discussed in greater detail hereinbelow.

Information entry device 16 provides an interactive interface with the user such as by a keyboard, modem, and/or other communication apparatus. Display 17 provides visual feedback to the user while output device 18 receives the results or the document production from data processor 10 in a conventional manner.

Fig. 2 presents a high level view of the present invention. The invention employs Structure Composer 20 to transform fragments of a structure from source 21 into a fully formed structure as output 22. The structure is defined through a Context Free Grammar (CFG) together with a set of context-dependent exceptions. The fully formed output structure 22 is legal according to the rules defined in the grammar 23. Missing portions of the structure are inferred by the invention. This inference is based upon a set of correction weights 24 which guide the inference process through the inference table 25.

The definition of the fully formed structure 22 is provided by a CFG together with a set of context-dependent exceptions determined by the grammar 23. The CFG portion of the grammar is preferably parseable using LL(k) parsing techniques. The exceptions are expressed as a list of inclusion exceptions and of exclusion exceptions that apply to each term in the grammar. An inclusion exception is a list of grammar terms that can appear anywhere in a sequence of terms derived from the term to which the inclusion exception is attached. An exclusion exception is a list of grammar terms that are forbidden from appearing in a sequence of terms derived from the term to which the exclusion exception is attached, even if the term would be allowed by the CFG portion of the grammar. The grammar can be defined through any traditional method of grammar definition, including Backus-Naur Form (BNF) and SGML Document Type Definition (SGML DTD). The grammar can contain cyclic rules such as those illustrated in Fig. 4 for lists.

The grammar 23 is considered to define the allowed structures in the following way: All terminals in the grammar represent actual content that can appear in the structure (such as text, graphics, computer programs, etc.). Non-terminals in the grammar represent containers providing additional logical structure around the actual content. A special type of container may contain both content and additional terminal or non-terminal children (such as text that contains a graphic). If the actual content was processed by an LL(k) parser, the logical structure corresponds to the parse tree constructed by such a parser. Each non-terminal corresponds to an interior node in this parse tree. In the logical structure, each non-terminal is considered to have two special markers in the structure: one marking where the non-terminal begins (the start tag) and one marking where it ends (the end tag). These additional markers are added to the set of terminals for the grammar and the non-terminal rules are rewritten to include these tags at the beginning and end of each rule.

The grammar definition 23 and the correction weights 24 are input to the Inference Table building process 25. This process uses standard parsing techniques to build a set of LL(k) non-recursive predictive parse tables for the grammar. See "Compilers, Principles, Techniques and Tools" by Aho, Sethi and Ullman, Addison-Wesley Publishing Co., for a description of these techniques. If the grammar was defined using an SGML DTD, see the text of ISO 8879 for a description of SGML DTDs. This information is also contained in "The SGML Handbook" by Charles F. Goldfarb. The DTD is converted to an LL(1) grammar specification as follows: each content model is rewritten as a BNF style production by introducing a new non-terminal representing the element and serving as the left-hand side of the production. The right hand side is written by introducing two new terminals (as described above): one for the start tag and one for the end tag. The rule begins with the start tag and follows with terms for all the elements and groups contained in the content model. As appropriate, repetition operators and connectors are expanded into multiple rules using standard grammar rewriting techniques. Any groups in the content model are represented using new non-terminals whose rules are constructed based on the contents of the group in the content model.

At the same time that the parse tables are built, the table build process analyzes the grammar and produces a directed graph representation of it which connects each symbol in the grammar with all other symbols which either directly derive the symbol or which are directly derived by the symbol (see Fig. 5 for an illustration of such a graph). This graph is used to quickly discover which symbols in the grammar can derive a given symbol during the inference process 20. Associated with the arcs in the graph are costs for traversing the arcs, i.e., costs for deriving the given symbol from the symbol at the other end of the arc. These costs are calculated based on the set of correction weights that provide rules for calculating the cost of creating any given terminal symbol. The cost of deriving a given symbol (the target symbol) from another symbol by following a particular production is the cost of producing the minimum set of required terminals to produce a legal parse context in which the target symbol can be derived from the original symbol. Finally, these minimum sets of terminals required to produce any given symbol from another symbol using a particular production are recorded and associated both with the particular production and with the arc connecting the two symbols in the graph.

The invention accepts as input fragments 21 of a fully formed instance 22 of the structure defined by the grammar 23. The fragments are presented to the invention sequentially. Each fragment is a sequence of terminals from the grammar together with a qualification value. The term qualification refers to a list of non-terminals together with two flags associated with each fragment. The list of non-terminals provide a derivation path to the top-level symbols in the fragment and serves to provide a context in which the fragment must fit. For instance, a title can be specified as a Section title as opposed to an Appendix title. The list of non-terminals can be empty, referred to as generic qualification, or it may be a complete set of non-terminals providing a derivation path from the start symbol of the grammar to the top-level symbols of the fragment, referred to as full qualification, or it may provide a partial derivation path (starting at some other symbol than the start symbol) to the top-level symbols of the fragment, referred to as partial qualification. As an example, TITLE is generically qualified, DOCUMENT.APPENDIX.TITLE is fully qualified, APPENDIX.TITLE is partially qualified.

The two flags included in the qualification are called "Start New" and "Forced Close". The Start New flag specifies one of the top-level non-terminals in the list of non-terminals providing the derivation path for the fragment. It indicates that the specified non-terminal must be "started" before the fragment can be created. In other words, the symbol specified by the Start New flag must be inferred even if that symbol is currently active. It serves to force the beginning of a new object such as a list or section. The Forced Close flag serves the opposite purpose. It indicates that a certain non-terminal should not be closed even if closing that non-terminal would allow the current fragment to be accommodated with the least-cost set of inferences. It serves to ensure that a particular object such as a list or section remains open.

The set of fragments presented to the invention is called the "stream of fragments". Note that the stream of fragments may be completely available when the process begins as in batch processing or may be dynamically generated as the process runs as occurs during interactive processing.

The output 22 of the invention is a fully formed, legal representation of the structure defined by the grammar 23. This structure can be represented as a hierarchy of objects. The input stream of fragments may represent only a portion of the fully formed structure. The invention infers the missing pieces of the structure completing the legal instance. The invention will fully qualify all fragments and insert all missing structural elements.

The missing portions of the fully formed legal structure are inferred by the structure composer 20. The process of inference is fully generalized in the invention. This process is controlled by the tables generated by the table building process 25 which are in turn based on the grammar 23 and a set of weights 24 used by the invention to calculate costs while structures are being inferred. These weights directly relate to the rules defined in the grammar 23.

Fig. 3 presents a detailed view of the present invention. The invention transforms an input stream with partial and incomplete structure from source 34 into an output stream 35 whose structure is complete. The output stream represents a legal instance as defined by the grammar 23. This transformation process is fully automated within the invention. Each fragment in the input stream is input to the system and results in one or more output fragments. The following paragraphs describe each stage of this transformation process.

The Qualification Manager (QM) 30 processes the qualification associated with each of the input fragments. The QM keeps track of the current state of the fully formed structure. This state includes both the parse stack used by the LL(k) parser 31 and high level information describing the actual structures currently active. The high level information includes a list of all the currently open structural elements together with a reference to the final term on the parse stack derived from that open element. (An element is considered open when one of the productions using it as the left-hand side has been processed. In that case, all the elements in the right-hand side of the production have been placed on the parse stack. The final element in the right-hand side represents the end-tag for the production. When that final element is processed, the original left-hand side element is considered closed.) When the end-tag is removed from the stack, the QM knows that a structural element is no longer active. In addition, the high level information tracks any inclusion and exclusion exceptions specified by open elements. Finally, the high level information includes markers of forced close flags on fragments that have been presented and marked with the "Forced Close" flag.

When an input fragment from source 34 is presented to the QM, it decides how much of the qualification list is currently active by looking at the high level information. It does this by comparing the non-terminals in the qualification list with the non-terminals currently known to be open. As long as the elements match, that portion of the qualification list is active and need not be inferred. The QM must consider if the fragment has the "Start New" flag set. If the start new flag is set on the fragment this is considered a mismatch between the current qualification and the desired qualification. Once there is a mismatch between the current qualification and the desired qualification, the QM infers new non-terminals. In other words, the input fragment specifies a context it must occur within.

For example, the qualification list for the fragment may specify that it can only occur inside a list in a chapter. If a chapter element is already open, that portion of the qualification list is active. However, the list may not be active and thus may need to be inferred. The QM will infer a new list before it processes the fragment. For each portion of the qualification of the object not currently active, the QM will create a new unqualified fragment. The current state is changed by passing these fragments to the LL(k) Parser 31.

The LL(k) Parser 31 is a standard LL(k) parser. The input to the LL(k) Parser is the unqualified fragments received from the QM 30. Each fragment is parsed against the Grammar 23 which defines the legal structures. If the fragment is legal based upon the current state, the current state is updated and the fragment is output to output creator 33. If the fragment is illegal based upon the current state (i.e., a parse error would occur upon encountering that fragment), the fragment is sent to Error Recovery 32 and the current state is not updated.

Error recovery 32 performs a breadth first search from the goal state represented by the illegal fragment back to the current state. This search discovers a set of terms from the grammar that, if they had been present originally, would change the current state such that the goal fragment was actually legal. The search works as follows.

The goal state term from the original illegal fragment is taken as the current search term. The set of possible paths is initialized to consist of a single path containing only the goal state term. This path is made the current path.

While the current search term from the current path is not found on the LL(k) parse stack, the set of possible terms from which it could be derived is found and put into a list of possible next terms. The set of terms is found by using the directed graph representation of the grammar contained in the grammar tables 25. Given the current term, the graph specifies all the terms that derive that term together with the production used in that derivation.

A set of possible paths is then built by adding each of the terms from the list of possible next terms to the current path. As these paths are built, the cost for the path is increased by the cost contained in the grammar tables 25 for traversing the arc that connected the current term to the possible next term. These paths are then added to the complete list of possible paths. This list is kept sorted by cost so that the least-cost path is always available. When all the paths have been updated and added to the complete list of paths, the new least-cost path is made the current path and the first term from that path is made the current search term. The search process then continues as described in the previous paragraph.

Eventually, the current search term will be found on the LL(k) parse stack. This means that a legal path leading to the original goal state term has been found and a cost associated with that path has been calculated. If the current search term is on the top of the stack, the calculated cost is complete. If the term is not on the top of the stack, then there are additional costs for using the path incurred by removing the terms on the stack that are above the current search term. These costs come in two forms: If the term to be removed is an end-tag, then the additional cost is just the cost for inferring the end-tag and can be obtained from the grammar tables 25. If the term to be removed is a non-terminal, then the additional cost is the cost incurred by inferring a minimum set of terminals required to satisfy that non-terminal. Again, this cost and the minimum set of non-terminals can be obtained from the grammar tables 25. The additional costs and any additional terminals required to remove non-terminals from the parse stack are added to the current path. If the current path is still the least cost path, the search is terminated. Otherwise, the new least cost path is made the current path and the search continues.

When a legal least cost path has been found (complete with any additional costs incurred as above), it is validated against the set of exclusions active on the parse stack up to the point where the current search term is found. If any of these exclusions specify terms in the least cost path, it is invalidated and the search continues. The legal least cost path also is validated so that when the path is parsed by the LL(k) parser, no terminal on the parse stack (end-tag) is removed that has been marked as "forced closed". The "forced closed" end tags must be explicitly presented to the invention. If a "forced closed" end tag is inferred in a legal least cost path, the path is invalidated and the search continues. If the exclusions do not invalidate the path, it is accepted and the tokens in it are returned to the LL(k) parser as described below. The parser will now undergo a series of changes to the current state that allow the originally illegal fragment to be accepted. This set of changes results in the minimal set of structural changes to accept the fragment.

To improve the performance of the breadth first search, heuristic pruning rules are used to limit the search space. There are two types of pruning depending on the fragment qualification of the current fragment being processed. If the qualification is generic, the search will only process the least cost path to any symbol in the grammar. Any path that leads to a higher cost symbol is not considered. If the qualification is fully or partially qualified, then the additional terms in the qualification list must be considered and paths which would close those terms must be rejected even if those paths have smaller costs than paths that preserve the terms in the qualification list. Any path that leads to an invalid qualification (i.e., would close terms on the qualification list or infer additional terms between terms on the qualification list) is not considered. Once a valid path is found it is recorded and the search continues until there are no other possible paths that cost less than the least-cost valid path.

If a valid path is found, the rules in the grammar are used to generate new fragments that complete the missing structures. These fragments are then parsed by the LL(k) Parser to transform the current state to the goal state. All fragments generated by error recovery are passed to the output creator 33. These fragments are generated in the process of the search. One or more structural elements may need to be inferred to construct a valid path.

When no valid path exists to a fragment, there are two possible courses of action. The action is controlled by the application that is feeding the fragments to the invention. The first is to conclude the current structure (i.e., produce the rest of the document), and the start of a new structure with this fragment as the first thing being placed inside it. The second is to "comment out" the fragment. In this case error recovery will mark the fragment as being "commented out" and leave the current state unchanged.

The output creator 33 buffers all of the fragments generated for each input fragment. Once an input fragment is processed, a stream of one or more output fragments is output from the output creator 33 to form the output stream 35.

While the preferred embodiments of the present invention are described herein with particularity, those having normal skill in the art will recognize various changes, modifications, additions and applications thereof other than those specifically mentioned herein without departing from the spirit of this invention.

What is claimed is:

CLAIMS

1. The method suitable for implementation in a computer environment for composing fragments of input data having partial markup and qualification information associated therewith into a structured organization comprising the steps of

selecting grammar rules defining the outline for the structured organization,

storing said selected grammar rules as tables and indices useful during structure completion from said selected grammar rules,

inspecting each input data fragment for modifications of structure within and around said fragment necessary for that fragment to match with said stored grammar rules, and completing the logical structure within and around said fragment in accordance with the results of said inspecting step

whereby a complete logical structure that is valid in accordance with said selected grammar rules is produced by said completing step.

2. The method in accordance with claim 1 wherein said inspecting step includes the step of discarding input data fragments having markup and qualification information associated therewith which are not compatible with said selected grammar rules.

3. The method in accordance with claim 2 wherein said completing step further includes the step of producing a complete document incorporating all received fragments which validly conform to said selected grammar rules.

4. The method in accordance with claim 1 wherein said inspecting step includes the step of inspecting each said received input data fragment, and

determining the minimal set of structural modifications within and around said fragment to the extent necessary for it to match the said stored grammar rules.

5. The method in accordance with claim 1 wherein said storing step further includes the step of building tables useful in conjunction with said inspecting step including the steps of

selecting a context free grammar along with context dependent exceptions, and

establishing a set of correction weights.

6. The method in accordance with claim 5 wherein said table building step includes the step of

constructing tables and indices including a first segment containing all possible derivation relations between symbols in said selected grammar, a second segment identifying minimum cost symbols that satisfy a given production, and a third portion specifying the minimum cost production sequence that allows derivation of a symbol from another symbol.

7. The method suitable for implementation in a computer environment for handling fragments of input data having partial markup and qualification information associated therewith and intended for composition into a structured organization determined by selected grammar rules which are stored as tables and indices useful during structure completion from said selected grammar rules comprising the steps of

sequentially receiving said input data fragments with their associated markup and qualification information,

inspecting each said received input data fragment, and

determining the minimal set of structural modifications within and around said fragment to the extent necessary for it to match the said stored grammar rules.

8. The method in accordance with claim 7 wherein said determining step further includes the step of conducting a breadth-first, cost-driven search from the symbols in said fragment to current document structures defined by said selected grammar rules through the space of all possible structures.

9. The method of building tables useful in conjunction with composing structured documents from a stream of data fragments each having markup and qualification information associated therewith comprising the steps of

selecting a context free grammar along with context dependent exceptions,

establishing a set of correction weights, and

constructing tables and indices including a first segment containing all possible derivation relations between symbols in said selected grammar, a second segment identifying minimum cost symbols that satisfy a given a given production, and a third portion specifying the minimum production sequence that allows derivation of a symbol from another symbol.

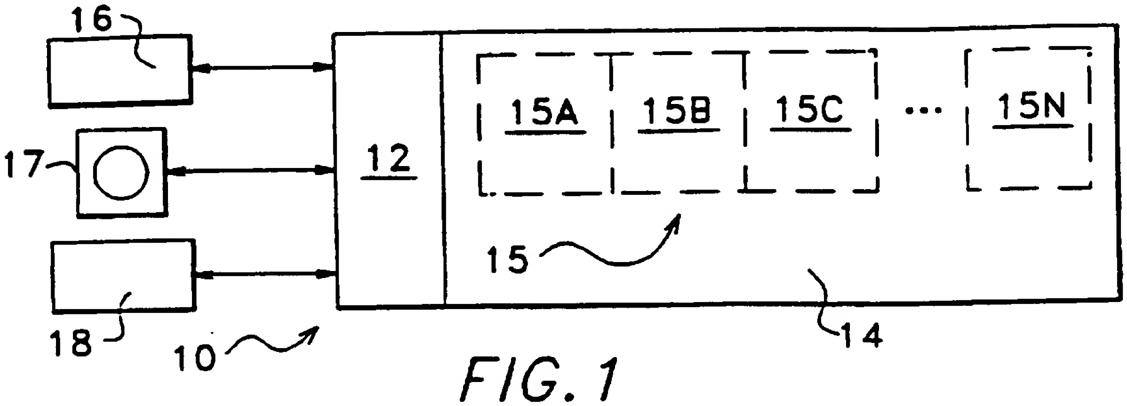


FIG. 1

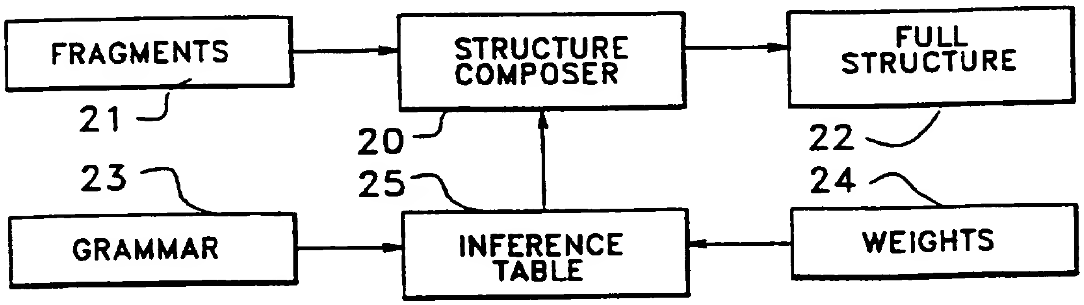


FIG. 2

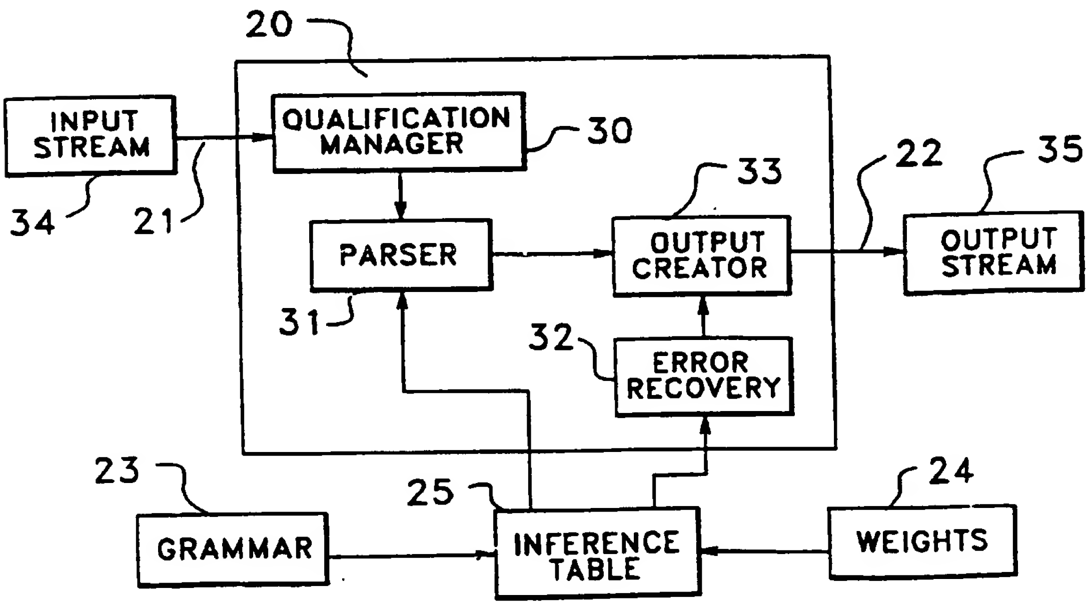


FIG. 3

FIG. 4

$\ell \rightarrow i+$

$i \rightarrow P+,\ \ell?$

$P \rightarrow \text{TEXT},\ |\ell$

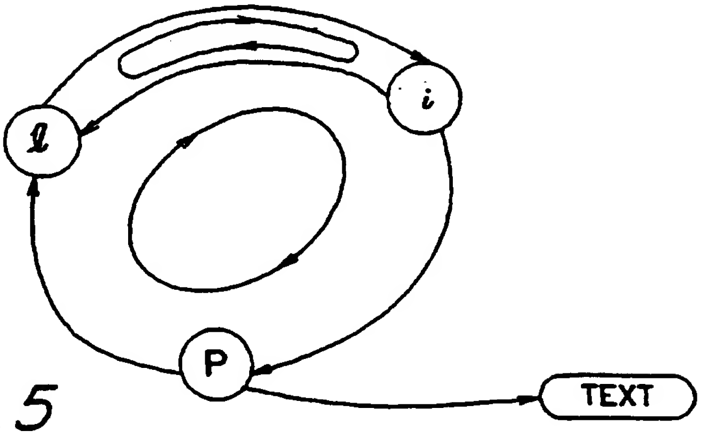


FIG. 5

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US95/15266

A. CLASSIFICATION OF SUBJECT MATTER

IPC(6) :G 06 F 17/22, 17/27
US CL :364/419.08; 395/145, 148
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 364/419.01, 419.1, 419.08; 395/144, 145, 146, 148, 375, 600

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

Please See Extra Sheet.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y,P	US, A, 5,438,512 (MANTHA ET AL) 01 August 1995, abstract, fig. 11; col. 1, line 23 to col. 4, line 40; col. 5, line 42 to col. 6, line 19; col. 9, line 52 to col. 11, line 35	1-9
A	US, A, 5,341,469 (ROSSBERG ET AL) 23 August 1994, col. 8, line 43 to col. 10, line 28; col. 24, line 46 to col. 26, line 19	5-6 & 9
A	US, A, 5,321,773 (KOPEC ET AL) 14 June 1994, col. 1, line 11 to col. 3, line 55; col. 19, lines 40-68; col. 25, line 13 to col. 26, line 64	1-9
A	US, A, 5,020,112 (CHOU) 28 MAY 1991, col. 1, line 10 to col. 3, line 2; col. 4, line 46 to col. 6, line 55	1-9

☒ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be part of particular relevance		
"E"	earlier document published on or after the international filing date	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed	"&"	document member of the same patent family

Date of the actual completion of the international search 02 FEBRUARY 1996	Date of mailing of the international search report 05 MAR 1996
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer ROBERT A. WEINHARDT Telephone No. (703) 305-9780

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US95/15266

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	Addison-Wesley Publishing Company, 1984, Patrick Henry Winston, "Artificial Intelligence, Second Edition", pp. 84-101	8

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US95/15266

B. FIELDS SEARCHED
Electronic data bases consulted (Name of data base and where practicable terms used):

APS

search terms: grammar, SGML, GML, layout/logical structure, context free grammar